Pointwise circumscription is equivalent to predicate completion (sometimes)

Yves Moinard

Pointwise circumscription is equivalent to predicate completion (sometimes)

Yves MOINARD

IRISA, Campus de Beaulieu, 35042 Rennes, France.

Abstract

Two ways of minimizing positive information are predicate completion, which comes from logic programming theory, and circumscription, which is one of the best known methods of non-monotonic reasoning. Here it is proved that, under certain hypotheses, there is equivalence between predicate completion and a special kind of circumscription.

I Introduction

The problem of "minimizing positive information" stems from data bases and logic programming theory and it has been addressed via various appoaches. Two of them are predicate completion and circumscription. It is logic programming which has given rise to predicate completion, and [6] deals with the tight connections existing between logic programming and circumscription.

Reiter has shown that, under certain circumstances, circumscription implies predicate completion [10]. Lifschitz notes that the converse is true when no recursive definition occurs [4]. Here we prove that a particular kind of circumscription, weaker than the general form, and called pointwise circumscription [5], is equivalent to predicate completion over the class of theories which basically correspond to logic programs, even if recursive definitions occur.

II Closed world assumption & negation as failure

Example 1: T: CUBE(a), [1]
CUBE(c),
CYLINDER(b),
ON(a,b).

From this, we would like to conclude that *b* is not a *cube*, in symbols ¬CUBE(b), which is not a logical consequence of the given formulas.

One solution to this problem is to give (directly or not) all the positive information, and to deal with negative information in a special way.
— Closed world assumption (CWA) [9]: for every ground instance Q(t) of a predicate Q, if Q(t) cannot be proved, then we infer ¬Q(t).
— Negation as (finite) failure (NAF) [2]: it is an effective method, using linear resolution for Horn (or "quasi Horn") clauses.

CWA, as expressed here, leads to inconsistency with some non Horn theories.
Example 2: T2: P(a) ∨ P(b). CWA gives ¬P(a) and ¬P(b).

NAF does not give all the ¬Q(t)'s obtained from CWA, even in cases where CWA is consistent (example 3 below is such a case).

III Predicate completion

It is a syntactic answer to the problem of the preceding section, see [2]. We consider here only programs which are finite sets of definite Horn clauses (one positive literal in each disjunction). For the sake of simplicity in notations, our predicates are unary (there is no special problem with other arities).

Let TE be such a program, and let the following be the clauses in TE with a positive literal in P.

$\forall y_1,\ldots,\forall y_{m_1} (C_1 \Rightarrow P(t_1)) \quad \ldots \quad \forall y_1,\ldots,\forall y_{m_n} (C_n \Rightarrow P(t_n))$, that is:

$\forall x\ (\exists y_1,\ldots,\exists y_{m_1}\ (x=t_1(y_1,\ldots,y_{m_1}) \wedge C_1) \Rightarrow P(x)), \ldots$
$\forall x\ (\exists y_1,\ldots,\exists y_{m_n}\ (x=t_n(y_1,\ldots,y_{m_n}) \wedge C_n) \Rightarrow P(x)).$
We write $E_i[x]$ for $\exists y_1,\ldots,\exists y_{m_i}\ (x=t_i(y_1,\ldots,y_{m_i}) \wedge C_i)\ \ (1 \leq i \leq n)$.
The conjunction of these formulas gives the *general form* of the *definition of* P *in* TE: $\forall x\ ((E_1[x] \vee \ldots \vee E_n[x]) \Rightarrow P(x))$.
The *completion* of predicate P in TE consists in adding the only-if ($\Leftarrow$) part in the definition of P, thus obtaining the following "implicit definition" of P in TE:

$$\forall x\ (\ P(x) \Leftrightarrow (E_1[x] \vee \ldots \vee E_n[x])\ ).$$

As we have introduced the equality symbol, we must add axioms for equality (including the substitutivity axiom schema).

In example 1, $\forall x(\ (x=a \vee x=c) \Rightarrow CUBE(x))$ is the general form of the definition of CUBE in T1, and then the completion of CUBE in T1 gives:

$$\forall x(\ CUBE(x) \Leftrightarrow (x=a \vee x=c)\ ).$$

Then we obtain: $(b \neq a \wedge b \neq c) \Rightarrow \neg CUBE(b)$.
NAF gives the desired negative answer: $\neg CUBE(b)$.
NAF is equivalent to completion provided we add "constant separation axioms" (also called Clark's axioms) to the theory (see [2]). Informally, these axioms state that two ground terms, if syntactically distinct, are not equal.

In example 1, these axioms give $a \neq b$, $a \neq c$, $b \neq c$ and then the completion of P in T1 gives $\neg CUBE(b)$.

Here is an example where completion is weaker than CWA.

Example 3: T3: $P(a) \Rightarrow P(b)$, $P(b) \Rightarrow P(a)$.
CWA is consistent (it is always consistent with Horn clauses) and gives $\neg P(a)$ and $\neg P(b)$.
The completion of P in T3 gives:
$T3 \wedge \forall x\ (\ ((x=a \wedge P(b)) \vee (x=b \wedge P(a))) \Leftrightarrow P(x)) \ \equiv$
$\forall x\ (\ P(x) \Leftrightarrow (\ P(a) \wedge (x=a \vee x=b)\ )).$
$\equiv$ is a metasymbol to be read: "is equivalent to".

That is, $P(a) \wedge P(b)$ is not ruled out by predicate completion of P in T3. It is the typical case were NAF goes into an infinite loop.

Predicate completion is of course strongly syntax-sensitive. Shepherdson gives the most striking example [11]:
We add a new clause to a program TE, $\forall x\ (P(x) \Rightarrow P(x))$, calling TE' the resulting program. The completion of P in TE' gives:
$\forall x\ (\ P(x) \Leftrightarrow (E_1[x] \vee \ldots \vee E_n[x] \vee P(x)\ )\ )$, that is:
$\forall x\ (\ (E_1[x] \vee \ldots \vee E_n[x]) \Rightarrow P(x)\ )$. So, we get TE before completion!

IV Standard predicate circumscription

McCarthy [7] proposes another approach to our minimization problem. Let T be a finite set of first order formulas in which the predicate P occurs. The (first order) *circumscription of* P *in* T consists in adding the following axiom schema to T:

$$\{T[p] \wedge \forall x\ (p[x] \Rightarrow P(x))\} \Rightarrow \forall x\ (P(x) \Rightarrow p[x]),$$

where p is any first order formula in the language of T. T[p] is T except that each occurrence of P is replaced by p, the notation p[x] means that p may have other variables than x (see [1] for details about added free variables).

We write Circum(T:P) for the union of T and these axioms.

Circum(T:P) is satisfied by those models of T in which the extension of P is minimal [7].

Let us return to example 1. The circumscription of CUBE in T1 gives the following instance of the axiom schema, associated with
$p[x] \equiv (x=a \vee x=c)$:

From T1, T1[p] is deducible, as it is T1 except that axioms 1 and 2 are replaced by $(a=a \vee a=c)$ and $(c=a \vee c=c)$ respectively.

$\forall x\ (p[x] \Rightarrow CUBE(x))$ is also deducible from T1. Then we get:

$\forall x\ (\ CUBE(x) \Rightarrow p[x]\ )$.

Circum(T1:CUBE) is equivalent to the completion of CUBE in T1.

In example 3, we use the instance associated with $p[x] \equiv$ FALSE, thus getting $\forall x\ (P(x) \Rightarrow \text{FALSE})$, that is $\forall x\ \neg P(x)$. Circum(T3:P) is stronger than predicate completion, and it is as strong as CWA.

Reiter proves that, for Horn theories (and even for theories "Horn in P"), circumscription implies completion [10]. Example 3 shows that the converse is false. However, Lifschitz notes that for a Horn theory in which the definition of P is $\forall x\ (E[x] \Rightarrow P(x))$, where E[x] is <u>without any occurrence of P</u>, then the converse is true [4].

V Pointwise circumscription

P is a predicate symbol, $P_{/y}$ is a notation for the formula defined by the following equivalence: $P_{/y}(x) \equiv (P(x) \wedge x \neq y)$.

Lifschitz [5] defines the *pointwise circumscription* of P in T that consists in adding to T the circumscription instance associated with $P_{/y}$, i.e.: $(T[P_{/y}] \Rightarrow \neg P(y))$.

—Indeed we have $\forall x\ (P_{/y}(x) \Rightarrow P(x))$

and $\{\forall x\ (P(x) \Rightarrow P_{/y}(x))\} \Leftrightarrow \neg P(y)$—

We get an open formula, that may be closed by $\forall$: $\forall y\ (\neg T[P_{/y}] \vee \neg P(y))$.

We write Cppp(T:P) for the union of T and this formula [2].

The main advantage is that there is no problem in finding "useful instances" of an axiom schema.

Clearly, Circ(T:P) entails Cppp(T:P). Also, if the theory T is positive in P, then Cppp(T:P) is equivalent to Circ(T:P) [5].

—A formula is *positive in P* if, not using "abbreviations" such as $\Rightarrow$ or $\Leftrightarrow$, every atom in P is in the scope of an even number of $\neg$'s, a formula *negative in P* is the negation of such a formula —.

It can be shown that, if F is a formula negative in P, then F need not to be taken into account for circumscriptions [8] :

Circum($F \wedge$T:P) $\equiv$ $F \wedge$Circum(T:P), and Cppp($F \wedge$T:P) $\equiv$ $F \wedge$Cppp(T:P).

So the formulas that are not negative in P are the only ones to be considered.

In example 1 (Horn and positive in CUBE), Cppp(T1:CUBE) is equivalent to Circum(T1:CUBE) and to the completion of CUBE :

$\mathrm{Cppp(T1{:}CUBE)} \equiv$

$\mathrm{T1} \wedge \forall x\, (\neg \mathrm{T1}[\mathrm{CUBE}/_x] \vee \neg \mathrm{CUBE}(x)) \equiv$

$\mathrm{T1} \wedge \forall x\, (\neg \mathrm{CUBE}(x) \vee \neg((\mathrm{CUBE}(a) \wedge a \neq x) \wedge (\mathrm{CUBE}(c) \wedge c \neq x))) \equiv$

$\mathrm{T1} \wedge \forall x\, (\mathrm{CUBE}(x) \Rightarrow ((\mathrm{CUBE}(a) \Rightarrow a = x) \vee (\mathrm{CUBE}(c) \Rightarrow c = x))) \equiv$

$\mathrm{T1} \wedge \forall x\, (\mathrm{CUBE}(x) \Rightarrow (x = a \vee x = c)).$

In example 3 (Horn but with recursivity in P), pointwise circumscription is equivalent to completion, but not to circumscription:

$\mathrm{Cppp(T3{:}P)} \equiv \mathrm{T3} \wedge \forall x\, (\neg P(x) \vee \neg((P(a) \wedge a \neq x) \Rightarrow (P(b) \wedge b \neq x))$
$\vee \neg((P(b) \wedge b \neq x) \Rightarrow (P(a) \wedge a \neq x))) \equiv$

$\mathrm{T3} \wedge \forall x\, (P(x) \Rightarrow ((P(a) \wedge a \neq x \wedge x = b) \vee (P(b) \wedge b \neq x \wedge x = a)) \equiv$

$\forall x\, (P(x) \Leftrightarrow (P(a) \wedge a \neq b \wedge (x = a \vee x = b))).$

So, if we add $a \neq b$ to T3, getting T'3, Cppp(T'3:P) is equivalent to the completion of P in T'3, and thus is weaker than Circum(T'3:P).

VI Pointwise circumscription & predicate completion

Now, we generalize these two examples. Let TE be as in III.

- The only formulas in TE non negative in P are the clauses written in III, which form the definition of P in TE. $\mathrm{Cppp(TE{:}P)} \equiv$

$\mathrm{TE} \wedge \forall y\, (P(y) \Rightarrow \neg \forall x\, ((E_1{}^y[x] \vee \ldots \vee E_n{}^y[x]) \Rightarrow (P(x) \wedge x \neq y))) \equiv$

$\mathrm{TE} \wedge \forall y\, (P(y) \Rightarrow \exists x\, ((E_1{}^y[x] \vee \ldots \vee E_n{}^y[x]) \wedge (\neg P(x) \vee x = y))).$

$E_i{}^y[x]$ is $E_i[x]$ where every occurrence of P is replaced by $P/_y$.

Since we consider only Horn clauses, $E_i[x]$ is positive in P, as well as the disjunction of the $E_i[x]$'s; then we get:

$(E_1{}^y[x] \vee \ldots \vee E_n{}^y[x]) \Rightarrow (E_1[x] \vee \ldots \vee E_n[x])$. According to TE we get:

$(E_1[x] \vee \ldots \vee E_n[x]) \Rightarrow P(x)$, hence:

$\mathrm{Cppp(TE{:}P)} \equiv \mathrm{TE} \wedge \forall y\, (P(y) \Rightarrow \exists x ((E_1{}^y[x] \vee \ldots \vee E_n{}^y[x]) \wedge (x = y))).$

This yields:

Theorem 1: $\mathrm{Cppp(TE{:}P)} \equiv \mathrm{TE} \wedge \forall y\, (P(y) \Rightarrow (E_1{}^y[y] \vee \ldots \vee E_n{}^y[y]))$ {Res}.

Corollary: Cppp(TE:P) implies the completion of P in TE.

This is a strengthening of Reiter's result: pointwise circumscription is sufficient to imply completion. Moreover we are very close to the converse.

If P does not occur in E_i, then E_i^y is E_i.

If P occurs in E_i, then $E_i[x]$ is $\exists \underline{z}\ (x=t_i[\underline{z}] \wedge \underline{P}(\underline{t}'_i[\underline{z}]) \wedge C'_i)$.

$\underline{z}$ denotes a sequence of free variables,

$\underline{P}(\underline{t}'_i[\underline{z}])$ denotes a conjunction $P(t'_{i_1}[\underline{z}]) \wedge \ldots \wedge P(t'_{i_{l_i}}[\underline{z}])$.

C'_i denotes a conjunction of positive literals without P.

$E_i^y[x] \equiv \exists \underline{z}\ (\ x=t_i[\underline{z}] \wedge \underline{P}(\underline{t}'_i[\underline{z}]) \wedge \underline{t}'_i[\underline{z}]\neq y \wedge C'_i)$.

$\underline{t'i}[\underline{z}]\neq y$ denotes the conjunction $t'_{i_1}[\underline{z}]\neq y \wedge \ldots \wedge t'_{i_{l_i}}[\underline{z}]\neq y$.

$E_i^y[y] \equiv \exists \underline{z}\ (\ y=t_i[\underline{z}] \wedge \underline{P}(\underline{t}'_i[\underline{z}]) \wedge \underline{t}'_i[\underline{z}]\neq y \wedge C'_i)$.

So, if TE implies $t'_{i_j}[\underline{z}] = t_i[\underline{z}]$, for some j, the corresponding $E_i^y[y]$ is FALSE and it vanishes in the disjunction of {Res}. Then a tautologous clause, such as $\forall x\ (P(x) \Rightarrow P(x))$, does not play any role in pointwise circumscription (Shepherdson's example shows that this is not the case with predicate completion).

Let us suppose that the program TE is a finite set of Horn clauses, without any *tautologous clause* of the kind $\forall z(\ (P(t[z]) \wedge \ldots) \Rightarrow P(t[z]))$, in a logical framework that includes Clark's axioms. In this case, for each clause in TE $\forall z(\ (P(t[z]) \wedge \ldots) \Rightarrow P(t'[z]))$, we get: $\forall z\ \ t[z]\neq t'[z]$, and then: $\forall y\ (E_i^y[y] \Leftrightarrow E_i[y])$. Thus, if we assume the completion formula $TE \wedge \forall y\ \ (P(y) \Rightarrow (E_1[y] \vee \ldots \vee E_n[y])$, we get the formula {Res}: $TE \wedge \forall y\ (P(y) \Rightarrow (E_1{}^y[y] \vee \ldots \vee E_n{}^y[y]))$.

We have established:

<u>Theorem 2:</u> If TE is a finite set of Horn clauses, without any tautologous clause, in a logical framework that includes Clark's axioms, then Cppp(TE:P) is equivalent to the completion of P in TE.

The same is true for predicates of any arity, and for tuples of predicates. Note that here we may have "recursive clauses" as well.

VII Discussion and conclusion:

This result is not very surprising, bearing in mind that, under the given hypotheses, predicate completion is equivalent to NAF. The axiom of pointwise circumscription is in fact a kind of NAF: if we cannot establish P(x), everything else staying identical, then we assume $\neg$P(x). Cases of minimization that do not fall under NAF (see example 3) are those cases where we should make an additional assumption: P(a) gets false if P(b) gets false also. Neither pointwise circumscription, nor NAF, can do this. Standard circumscription, and some extended NAF (detecting infinite loops) can do this.

Pointwise circumscription can be considered as a good way of extending the notion of predicate completion to any theory. The price is that pointwise circumscription is no longer syntax-dependant (it is only "vocabulary dependant", as any circumscription). So, oriented non Horn clauses as they are considered in [2], loose their "orientation" (there is no longer any privileged litteral to be written on the right part of $\Rightarrow$ in the implicative form of a clause). One gain is that tautologous formulas are no longer a trouble.

Acknowledgements

I would like to thank Philippe Besnard for his judicious advices.
I also thank several referees for usefull comments and amendments.

This research was sponsored by CNRS PRC GRECO IA and INRIA.

Notes

1 The examples given are standard ones, chosen as being appropriate for the purposes of this paper.

2 A referee pointed out that in the propositional case, $P_{/y}$ is FALSE, thus pointwise circumscription has a meaning in this case also.

References

[1] P. Besnard, R. Mercer, Y. Moinard: *The importance of open and recursive circumscription.* Submitted for publication.

[2] K. L. Clark: *Negation as failure*, in [3].

[3] H. Gallaire and J. Minker editors: Logic and Data Bases. Plenum, New York 1978.

[4] V. Lifschitz: *Computing Circumscription.* Proc. IJCAI 1985, Los Angeles, pp.121-127.

[5] V. Lifschitz: *Pointwise Circumscription: Preliminary Report.* Proc. AAAI 1986, Philadelphia, p.406-410.

[6] V. Lifschitz: *On the Declarative Knowledge of Logic Programs with Negation*, in Foundations of deductive databases and logic programming, J. Minker ed., Morgan-Kaufmann, 1988; pp.177-191.

[7] J. McCarthy: *Circumscription —a form of non-monotonic reasoning.* Artificial Intelligence n°13, 1980, pp.27-39.

[8] Y. Moinard: *Contribution à l'étude de la circonscription.* Thesis, Rennes 1988.

[9] R. Reiter: *On Closed World Data Bases*, in [3].

[10] R. Reiter: *Circumscription implies predicate completion (sometimes).* Proc. AAAI 1982, Pittsburgh, pp.418-420.

[11] J. C. Shepherdson: *Negation as failure:...*. Journal of Logic Programming n°1, 1984, pp.51-79.